

Chapter 8: Enstore Commands

Enstore provides commands that allow you to communicate with various components of the system. The basic syntax of all Enstore commands is

```
% enstore <command> [--option [argument] ...]
```

All options start with a double dash (--).

8.1 enstore file

This command communicates with the File Clerk (see section 7.1 *File Clerk*). It returns information about a specified file or files on a specified volume.

Syntax:

```
% enstore file [--option [argument] ... ]
```

Options:

-h, --help	prints the options (i.e., prints this message)
------------	--

Example:

```
$ enstore file --help
```

```
Usage:
```

```
file [ -h --bfid= --help --list= --ls-active= --usage ]
```

```
--bfid <BFID>      get info of a file
-h, --help         print this message
--list <VOLUME_NAME> list the files in a volume
--ls-active <VOLUME_NAME> list active files in a volume
--usage           print short help message
```

--bfid <BFID>

Returns information (metadata) about the file corresponding to the specified bfid.

You can get the bfid of a file from the **enstore pnfs --bfid <FILE_NAME>** command (section 8.5 *enstore pnfs*); get the filename from searching PNFS namespace.

Example:

```
$ enstore file --bfid CDMS105770745000000
```

```
{'bfid': 'CDMS105770745000000',
 'complete_crc': 1191066979L,
 'deleted': 'no',
 'drive': 'stkenmvr7a:/dev/rmt/tps0dln:4560000022',
 'external_label': 'VO3222',
 'location_cookie': '0000_000000000_0005661',
 'pnfs_mapname': '',
 'pnfs_name0': '/pnfs/fs/usr/test/xyz/srmttest/ar017983.0001phys_10',
 'pnfsid': '000500000000000000190EA8',
 'pnfsvid': '',
 'sanity_cookie': (65536L, 3203712884L),
 'size': 197354833L}
```

--list <VOLUME_NAME>

Lists the files in the specified volume with their volume name, bfid, size, location (file number) on volume, delete flag, and the original filename in pnfs.

You can get the volume name from the **enstore pnfs** command, using either **--xref** or **--layer** (section 8.5 *enstore pnfs*), or from the “external_label” field of the **enstore file --bfid <BFID>** command (shown above).

The **enstore volume --list <VOLUME_NAME>** is an alias for this command.

Example:

```
$ enstore file --list VO3222
```

```
label bfid          size  location_cookie  delflag
original_name

VO3222 CDMS106503213600000 983803 0000_000000000_0011536 deleted
/pnfs/fs/usr/eagle/dcache-tests/274.dcache_page_p_27750
```

(This shows one of many lines appearing in the real output, and is reformatted to two lines for readability.)

```
--ls-active <VOLUME_NAME>
```

Lists active files in a volume.

You can get the volume name from the **enstore pnfs** command, using either **--xref** or **--layer** (section 8.5 *enstore pnfs*), or from the “external_label” field of the **enstore file --bfid <BFID>** command (shown above).

Example:

```
$ enstore file --ls-active VO3222
```

```
/pnfs/fs/usr/eagle/dcache-tests/101.dcache_page_a_24401
/pnfs/fs/usr/eagle/dcache-tests/101.dcache_page_24401
/pnfs/fs/usr/test/stress-test/myfile1
/pnfs/fs/usr/test/stress-test/myfile3
/pnfs/fs/usr/test/stress-test/file128m-11
...
```

```
--usage           prints short help message
```

Example:

```
$ enstore file --usage
```

```
Usage:
file [ -h --bfid= --help --list= --ls-active= --usage ]
```

8.2 enstore library

This command communicates with the Library Manager (see section 7.3 *Library Manager*). You can use it to get information pertaining to a particular Library Manager. Use the online monitoring pages (see Chapter 9: *Monitoring Enstore on the Web*) to find the library manager of interest.

Syntax:

```
% enstore library [--option [argument] ... ] <library>
```

The **<library>** argument is required except when using the **--help** option; the “**.library_manager**” portion of the library name is optional.

Options:

-h, --help prints this message (i.e., prints the options)

Example:

```
$ enstore library --help
```

```
Usage:
```

```
library [OPTIONS]... library
```

```
--get-asserts <library> print sorted lists of pending volume asserts
```

```
--get-queue <HOST_NAME> print queue submitted from the specified host.
```

```
    If empty string specified, print the whole queue
```

```
--get-suspect-vols      print suspect volume list
```

```
--get-work-sorted      print sorted lists of pending and active requests
```

```
-h, --help              prints this message
```

```
--usage                prints short help message
```

--get-asserts <LIBRARY>

prints sorted lists of pending volume asserts for specified library.

Example:

```
$ enstore library --get-asserts 9940.library_manager
```

```
Pending assert requests: 0
```

```
Active assert requests: 0
```

```
{'status': ('ok', None)}
```

--get-queue <HOST_NAME> <LIBRARY>

Prints queue submitted from the specified encp client host. Both arguments are required. If quoted empty string is specified for host name, it prints the whole queue (for all hosts).

Examples:

```
$ enstore library --get-queue stkensrv3 9940.library_manager
```

```
Pending write requests
Active requests
Pending read requests: 0
Pending write requests: 2
Active read requests: 0
Active write requests: 0
{'status': ('ok', None)}
```

The top two lines tell us that there are no pending or active transfers involving stkensrv3 for the 9940 library manager. The 4th line tells us there are 2 pending write requests for this library manager from hosts other than stkensrv3.

If all hosts are specified (the next example), the command returns the fields: host name, library manager, username (of encp request), input filename, and output filename for each pending and/or active request (3 shown here), and ends with a summary:

```
$ enstore library --get-queue "" 9940.library_manager
```

```
Active requests
fnsimu2 9940.library_manager lixn
  /pnfs/btev/geant2003/xiaonan/dstar_xiaonan_1.evt.gz
  /scr/bphys6/lixn/dstar_xiaonan_1.evt.gz M 9944
fsgi01 9940.library_manager rschultz
  /usr/bdms/rschultz/fl_066_uplsr7/fl_ed_066_uplsr7.ldhi
  /pnfs/BDMS/lens/fl_066_uplsr7/fl_ed_0663
fnsfh 9940.library_manager minfarm
  /export/stage02_minos/C00040259_0000.tdaq.root
  /pnfs/minos/caldet_reco/tdaq_data/2002-09/C0004027
Pending read requests: 0
Pending write requests: 0
Active read requests: 1
Active write requests: 2
{'status': ('ok', None)}
```

--get-suspect-vols <LIBRARY>

Prints suspect volume list for specified library manager.

Example:

```
$ enstore library --get-suspect-vols 9940.library_manager
```

```
[{'movers': ['994071.mover'], 'external_label': 'V04523',
'time': 1067290586.907726}, {'movers': ['994051.mover', '994061.mover', '']}]
```

```
--get-work-sorted <LIBRARY>
```

Prints sorted lists of pending and active requests.
It sorts by queue.

Example:

```
$ enstore library --get-work-sorted 9940.library_manager
{'write_queue': [], 'read_queue': [], 'admin_queue': []}
[{'status': ('ok', None), 'vc': {'status': ('ok', None),
'declared': 1011741604.130481, 'si_time': [1041612783.99499, 0], 'blocksiz
```

8.3 enstore monitor

This command communicates with the Monitor Server (see Chapter 9: *Monitoring Enstore on the Web*) to get network speed information.

On machines with an `enstore.conf` file (see Appendix A: *Network Control*), the `enstore monitor` command uses the routing already established there. If `enstore monitor` set up its own, it would interfere with the routes currently in use.

Syntax:

```
% enstore monitor [--option [argument] ...]
```

```
-h, --help          Prints this message (i.e., prints the options)
```

Example:

```
$ enstore monitor -h
Usage:
  monitor [ -h --help --host= --usage --verbose= ]

-h, --help          prints this message
--host <HOSTIP>    selects a single host
--usage            prints short help message
--verbose <VERBOSE> print out information.
```

```
--host [HOST_NAME or IP_ADDRESS]
```

Returns network rate for the specified host (Enstore node). If you don't specify host, it runs the command for all hosts. Example below shows results for a single host.

Example:

```
$ enstore monitor --host stkensrv3
Trying stkensrv3.fnal.gov
Network rate measured at 11.33 MB/S recieving and 11.1 MB/S sending.
```

`--verbose <INTEGER_VALUE>`

This command is used to help find and fix network problems. It prints detailed information about actions taken. The higher the number you give as an argument, the more info displayed.

Example:

```
$ enstore monitor --host stkensrv3 --verbose 20
```

```
6 Tue Oct 28 10:48:13 2003 msc called with args: ['monitor', '--host',
'stkensrv3', '--verbose=20']
13 Tue Oct 28 10:48:13 2003 Get monitor_server config info from server
Trying stkensrv0.fnal.gov
13 Tue Oct 28 10:48:13 2003 Get None config info from server
13 Tue Oct 28 10:48:13 2003 Get None config info from server
13 Tue Oct 28 10:48:13 2003 Get log_server config info from server
13 Tue Oct 28 10:48:13 2003 Get log_server config info from server
13 Tue Oct 28 10:48:13 2003 Get None config info from server
13 Tue Oct 28 10:48:13 2003 Get alarm_server config info from server
...
10 Tue Oct 28 10:48:14 2003 Connecting to monitor server.
10 Tue Oct 28 10:48:14 2003 Obtaining error status for data socket.
10 Tue Oct 28 10:48:15 2003 Get the final dialog rate information.
Network rate measured at 11.34 MB/S recieving and 11.23 MB/S sending.
```

8.4 enstore volume

This command communicates with the Volume Clerk (see section 7.2 *Volume Clerk*) to return information on data volumes.

Syntax:

```
% enstore volume [--option [argument] ... ]
```

-h, --help Prints this message (i.e., prints the options)

Example:

```
$ enstore volume --help
```

```
Usage:
  volume [OPTIONS]...

  --gvol <VOLUME_NAME> get info of a volume in human readable time
                        format
  -h, --help            prints this message
  --just <VOLUME_NAME> used with --pvols to list problem
  --list <VOLUME_NAME> list the files in a volume
  --ls-active <VOLUME_NAME> list active files in a volume
  --ls-sg-count         list all sg counts
  --pvols              list all problem volumes
  --usage              prints short help message
  --vol <VOLUME_NAME> get info of a volume
  --vols              list all volumes
```

--gvol <VOLUME_NAME>

This is just like **enstore volume --vol <VOLUME_NAME>**, except that this one prints human-readable time fields (e.g., “declared”, “first_access” and “last_access” fields).

Example:

```
$ enstore volume --gvol VO3332
```

```
{'blocksize': 131072,
 'capacity_bytes': 64424509440L,
 'declared': 'Wed Jan 16 16:13:57 2002',
 'eod_cookie': '0000_0000000000_0000044',
 'external_label': 'VO3332',
 'first_access': 'Fri May 10 12:59:35 2002',
 'last_access': 'Mon Oct 27 22:35:45 2003',
 'library': '9940',
 'media_type': '9940',
 'non_del_files': 43,
 'remaining_bytes': 1785262080L,
 'sum_mounts': 234,
 'sum_rd_access': 213,
 'sum_rd_err': 0,
 'sum_wr_access': 43,
 'sum_wr_err': 0,
 'system_inhibit': ['none', 'full'],
 'user_inhibit': ['none', 'none'],
 'volume_family': 'cms.objy_data_files.cpio_odc',
 'wrapper': 'cpio_odc'}
```

--list <VOLUME_NAME>

This is an alias for the **enstore file --list <VOLUME_NAME>** command. See section 8.1 *enstore file*.

--ls-active <VOLUME_NAME>

lists original file names of active files in a volume

Example:

\$ enstore volume --ls-active VO3332

```
/pnfs/cms/UserFederation/data/jetmet_production/data/Collections/jm_Hit601_g125_UCSD/jm02_qqh120_11/EVD0.jet0102.DB
/pnfs/cms/UserFederation/data/jetmet_production/data/TAssoc/jm_2x1033PUjm602_TkMu_g125_UCSD/jm02_hlt15-20/EVD11.jet0102.DB
/pnfs/cms/UserFederation/data/jetmet_production/data/Digis/jm_2x1033PUjm602_TkMu_g125_UCSD/jm02_hlt0-15/EVD12.jet0102.DB
/pnfs/cms/UserFederation/data/jetmet_production/data/Hits/jm_Hit601_g125_UCSD/jm02_hlt230-300/EVD12.jet0102.DB
...
```

--ls-sg-count <VOLUME_NAME>

lists allocated tape counts by library and by storage group. If “storage group” has value “none”, the negative number under “allocated” gives the number of tapes that are available in the robot, but not yet assigned to a storage group.

Example:

\$ enstore volume --ls-sg-count VO3332

```
library      storage group  allocated
=====
...
9940         ktev          189
9940         lqcd          150
9940         miniboone     132
9940         minos         109
9940         none         -13
9940         patriot      20
9940         sdss         608
9940         test         28
9940         theory       70
CD-9940B     cms           129
...
```

```
--pvols [--just <VOLUME_1> <VOLUME_2> ...]
```

Without **--just**, this lists all problem volumes.
With **--just** followed by a space-separated list of volume names, it lists only the problem volumes among the given list.

The columns returned are: volume name, primary status, primary status time, secondary status, secondary status time. (The time fields are relatively new; not all volumes will display them.)

Example:

```
$ enstore volume --pvols
```

```
==== readonly
LEGL10      none      *      readonly 0913-1540
LEGL98      none      *      readonly 0819-2329
...
==== full
...
VO4845      none      *      full      *
VO4846      none 1023-1032  full      *
VO4847      none      *      full      *
VO4848      none      *      full      *
VO4849      none      *      full      *
VO4850      none      *      full 1016-2315
VO4851      none      *      full 1017-0409
...
$ enstore volume --pvols --just VO3332
(no sample output available)
```

`--vol <VOLUME_NAME>`

returns detailed information about specified volume

Example:

```
$ enstore volume --vol VO3332
```

```
{'blocksize': 131072,
'capacity_bytes': 64424509440L,
'declared': 1011219237.849051,
'eod_cookie': '0000_000000000_0000044',
'external_label': 'VO3332',
'first_access': 1021053575.259737,
'last_access': 1067315745.238969,
'library': '9940',
'media_type': '9940',
'non_del_files': 43,
'remaining_bytes': 1785262080L,
'sum_mounts': 234,
'sum_rd_access': 213,
'sum_rd_err': 0,
'sum_wr_access': 43,
'sum_wr_err': 0,
'system_inhibit': ['none', 'full'],
'user_inhibit': ['none', 'none'],
'volume_family': 'cms.objy_data_files.cpio_odc',
'wrapper': 'cpio_odc'}
```

`--vols`

lists all volumes with their available space, the system inhibits, the library, the volume family (period-separated concatenation of storage group, file family and file family wrapper) and any comments.

Example:

```
$ enstore volume --vols
```

label	avail.	system_inhibit	library	vol_family	comment
...					
VO0053	1.19GB	(none full)	eagle	cms.objy_data_files.cpio_odc	
VO0054	0.51GB	(none full)	eagle	cms.objy_data_files.cpio_odc	
VO0055	0.17GB	(none full)	eagle	theory.theory-canopy-C.cpio_odc	
VO0056	0.65GB	(none full)	eagle	theory.theory-canopy-D.cpio_odc	
...					

8.5 enstore pnfs

Enstore has a **pnfs** command that allows you to perform a variety of pnfs manipulations, as listed in the option table below. Off-site users cannot mount `/pnfs`, and therefore cannot run this command.

Syntax:



% enstore pnfs [--option [argument] ...]

Note that the options that reset pnfs tag values should be used only by your experiment's designated Enstore guru(s).

<pre> --help List the options for the enstore pnfs command: Example: % enstore pnfs --help Usage: pnfs [OPTIONS]... --bfid <FILENAME> lists the bit file id for file --cat <FILENAME> [LAYER] see --layer --file-family [FILE_FAMILY] gets file family tag, default; sets file family tag, optional --file-family-width [FILE_FAMILY_WIDTH] gets file family width tag, default; sets file family tag, optional --file-family-wrapper [FILE_FAMILY_WRAPPER] gets file family width tag, default; sets file family tag, optional --filesize <FILE> print out real filesize -h, --help prints this message --info <FILENAME> see --xref --layer <FILENAME> [LAYER] lists the layer of the file --library [LIBRARY] gets library tag, default; sets library tag, optional --tag <TAG> [DIRECTORY] lists the tag of the directory --tagchmod <PERMISSIONS> <TAG> changes the permissions for the tag; use UNIX chmod style permissions --tagchown <OWNER> <TAG> changes the ownership for the tag; OWNER can be 'owner' or 'owner.group' --tags [DIRECTORY] lists tag values and permissions --usage prints short help message --xref <FILENAME> lists the cross reference data for file </pre>
<pre> --bfid <FILE_NAME> returns the BFID of the file; select file name to specify from within pnfs space and use relative/absolute path as needed. Example: \$ enstore pnfs --bfid /pnfs/mist/zuu/100MB_002 WAMS104102942800000 </pre>
<pre> --cat <PATH_TO_FILE> [LAYER] --cat is an alias for --layer; see --layer. </pre>
<pre> --file-family prints the file family name associated with the current pnfs directory. Example: \$ enstore pnfs --file-family dcache </pre>



--file-family <FILE_FAMILY_NAME>

If permissions allow, this will reset the file-family name of the current pnfs directory to the specified value. This is for Enstore admins or designated gurus only (succeeds only if permissions allow).

Example:

```
$ enstore pnfs --file-family newfamilyname
```

--file-family-width

Prints the file family width associated with the current pnfs directory.

Example:

```
$ enstore --file-family-width  
1
```



--file-family-width <FILE_FAMILY_WIDTH>

If permissions allow, this will reset the file-family width of the current pnfs directory to the specified value. This is for Enstore admins or designated gurus only (succeeds only if permissions allow).

Example:

```
$ enstore --file-family-width 2
```

--file-family-wrapper

Prints the file family wrapper associated with the current pnfs directory.

Example:

```
$ enstore pnfs --file-family-wrapper  
cpio_odc
```



--file-family-wrapper <FILE_FAMILY_WRAPPER>

If permissions allow, this will reset the file-family wrapper of the current directory to the specified value. This is for Enstore admins or designated gurus only (succeeds only if permissions allow).

Example:

```
$ enstore pnfs --file-family-wrapper cpio_odc
```

`--filesize <PATH_TO_FILE>`

Prints the real filesize in bytes; useful for files of size greater than (2G-1) bytes, since PNFS stores file size as 1 in this case.

Example:

```
$ enstore pnfs --filesize a01
```

```
24198
```

`--info <PATH_TO_FILE>`

Prints information about the file, this is an alias for the `--xref` option. See `--xref`.

`--layer <PATH-TO-FILE> <LAYER>`

prints information about the file. Layer 0 is used internally by pnfs and it can't be viewed. Layer 1, the default, gives the file's BFID. Layer 2 is used by dCache. Layers 3, 5, 6, 7 are not currently used. Layer 4 produces output equivalent to `--xref`, but returns info without field labels.

The option `--cat` is an alias for this option.

Examples:

Layer 1 gives BFID (default):

```
$ enstore pnfs --layer a01
```

```
CDMS105889726300000
```

```
$ enstore pnfs --layer a01 1
```

```
CDMS105889726300000
```

Layer 2 is used for dCache:

```
$ enstore pnfs --layer a01 2
```

```
2,0,0,0.0,0.0
:c=1:d15ef6a3;l=554423;
w-fcdfdata018-1
```

The file has a version1 crc of `c=1:d15ef6a3`, it has a length `l=554423`, and it is in pool `w-fcdfdata018-1`.

```
$ enstore pnfs --layer a01 2
```

```
2,0,0,0.0,0.0
:
```

Layer 4 gives `--xref` output (see `--xref`):

```
$ enstore pnfs --layer a01 4
```

```
VO3222
0000_0000000000_0006264
24198
dcache
/pnfs/fs/usr/test/xyz/srmtest/test_1_1/a01

00050000000000000000191030

CDMS105889726300000
stkenmvr5a:/dev/rmt/tps3dln:4560000022
```

--tags [DIRECTORY] List the tag values of specified PNFS directory (if no directory argument, it lists tags for current working directory (cwd or pwd))

Example:

```
$ pwd
```

```
  /pnfs/test/xyz/srmtest/test_1_1
```

```
$ enstore pnfs --tags
```

```
.(tag)(file_family) = dcache
.(tag)(file_family_width) = 1
.(tag)(file_family_wrapper) = cpio_odc
.(tag)(library) = 9940
.(tag)(storage_group) = test
-rw-rw-r--  11 root    sys           6 Jul 26  2001
              /pnfs/test/xyz/srmtest/test_1_1/.(tag)(file_family)
-rw-rw-r--  11 root    sys           1 May  5  2000
              /pnfs/test/xyz/srmtest/test_1_1/.(tag)(file_family_width)
-rw-rw-r--  11 root    sys           8 May  5  2000
              /pnfs/test/xyz/srmtest/test_1_1/.(tag)(file_family_wrapper)
-rw-rw-r--  11 root    sys           4 Jul  3 10:59
              /pnfs/test/xyz/srmtest/test_1_1/.(tag)(library)
-rw-r--r--  11 root    sys           4 Jul 26  2001
              /pnfs/test/xyz/srmtest/test_1_1/.(tag)(storage_group)
```

(minor reformatting done to enhance readability)

`--xref <FILE_NAME>`List cross-reference information (metadata) for specified file. (`--info` is an alias for `--xref`.) The information includes:

- `volume`: storage media volume label
- `location_cookie`: file position on tape (the number of the file on tape)
- `size`: file size in bytes
- `file_family`: file family
- `original_name`: original name in `/pnfs` before any move/copy command issued; i.e., the destination filename given in the `encp` command used to copy the file to Enstore
- `map_file`: obsolete, but some older files may have a value here
- `pnfsid_file`: unique id for the file as assigned by PNFS
- `pnfsid_map`: obsolete, but some older files may have a value here
- `bfid`: unique id for the file as assigned by Enstore (matches layer 1)
- `origdrive`: id of drive used when file was written to media (files generated prior to 10/2000, `encp` v2.5 or earlier, will not have a value here)
- `crc`: CRC of the file (appears for files after 10/2003, using `encp` v3_1 or greater)

Example:

```
$ enstore pnfs --xref a01
```

```
volume: VO3222
location_cookie: 0000_0000000000_0006264
size: 24198
file_family: dcache
original_name: /pnfs/fs/usr/test/xyz/srmtest/test_1_1/a01
map_file:
pnfsid_file: 0005000000000000000191030
pnfsid_map:
bfid: CDMS105889726300000
origdrive: stkenmvr5a:/dev/rmt/tps3d1n:4560000022
crc: unknown
```

`--library` Returns the value of the library tag (the virtual library associated with files in the directory) for the current `pnfs` directory.

```
$ enstore pnfs --library
```

```
9940
```



`--library <LIBRARY>`

provides a name to reset the library tag. This is for Enstore admins or designated gurus only (succeeds only if permissions allow).

`$ enstore pnfs --library mylib`

`[Errno 13] Permission denied: '/pnfs/test/xyz/srctest/test_1_1/.(tag)(library)'`